Design and Standardization of Low-Density Parity-Check Codes for Space Applications

Kenneth Andrews, Dariush Divsalar, Sam Dolinar, Jon Hamkins, and Fabrizio Pollara*

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109, USA

In the last decade, a family of 16 low-rate turbo codes were designed and standardized, and decoders were built for each of the three ground complexes of the Deep Space Network (DSN). These codes are now flying on several missions, including MRO, MESSENGER, and STEREO. A brief summary of the turbo code family is presented, along with a performance and complexity comparison to NASA's legacy codes.

In the last few years, exciting research developments have produced ten higher rate low-density parity-check (LDPC) codes for the high-data-rate missions anticipated in the coming decades. Originally invented by R. Gallager in his PhD thesis in 1960, this coding technique was largely forgotten for more than 30 years. The primary advance in LDPC codes is the discovery of an iterative decoding algorithm, now called Belief Propagation (BP) decoding, which offers near-optimum performance for large linear LDPC codes at a manageable complexity. The performance gains of LDPC codes were difficult to realize technologically in the early 1960s. Several decades of VLSI development have finally made the implementation of these codes practical.

The renaissance of LDPC codes did not mark the end of turbo codes, however. LDPC codes have performance and complexity advantages over turbo codes at high code rates, but turbo codes are currently still the best solution for the lower code rates. This natural partition means that the standard family of turbo codes at rates 1/6, 1/4, 1/3, and 1/2 can live in harmony with a proposed standard of LDPC codes at rates 1/2, 2/3, 4/5, and 7/8. LDPC codes hold the promise of significantly lower decoding complexity and error-floors than turbo codes, while still achieving near-capacity performance. A summary is presented of the performance and complexity of the LDPC code family and related codes developed by NASA, along with a description of recent FPGA implementations at speeds in excess of 100 Msps. Synchronization aspects at low SNR are also discussed, and the status of infusion of this new technology in NASA missions is addressed.

Standardization of LDPC codes by the CCSDS has proven challenging. In part, this is because the class of LDPC codes is very large, and their differences can be small. There is also a long list of desirable attributes, and no code is superior in all categories. The status of the standardization process is described, including a discussion of the advantages and disadvantages of the proposed candidates.

I. Introduction

Forward error correction (FEC) has been an important component of space communication since the 1960's, when Reed-Muller and convolutional codes were used on early missions such as Pioneer and Mariner.[1] These early codes were decoded with majority vote logic and sequential decoding algorithms whose complexity was fairly modest and matched the available digital technology of the day.

As time went on, higher and higher data rates were requested from places farther and farther away. While the most straightforward way to meet these goals was to fly larger power amplifiers and use bigger antennas — and indeed NASA did just this when it extended the Deep Space Network (DSN) 64m antennas to 70m in 1982-1988 to service Voyager as it made encounters with Uranus and Neptune — perhaps the single most cost effective means to improve performance was the development of new FEC codes.

*The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

These advancements included the development of a complex maximum likelihood sequence (Viterbi) decoder for a constraint-length 15 convolutional code,[2] and the development of concatenated Reed-Solomon (RS) and convolutional codes. The concatenated code provided about 2 dB of additional coding gain over the NASA standard constraint-length 7, rate 1/2 convolutional code, putting the codes within about 2.5 dB of the Shannon limit.

Since 1960, many nations joined in the exploration of space. There was a need for standards to govern space data systems, because interoperable spacecraft, instruments, and ground support systems would encourage international collaboration in space exploration. Such interoperability would enable one space agency to provide cross support of data handling services for another agency, saving money for the nations' space agencies.

In response to this emerging need, in 1982 the Consultative Committee for Space Data Systems (CCSDS) was formed. Its charter is to identify elements of space communications systems that can be standardized, and to develop consensus recommendations for such elements, so that interoperability can be maximized. It comprises ten member space agencies, twenty observer space agencies, and more than a hundred private company associates.

Under this standards-based umbrella, development of FEC codes continued. When turbo codes broke onto the scene in 1993, the Jet Propulsion Laboratory developed a family of codes of rates 1/6, 1/4, 1/3, and 1/2, and blocklengths 1, 2, 4, and 5 times a base length of 1784 information bits. This family of codes was standardized by CCSDS in 1999,[3] and is now used by several spacecraft including MESSENGER (a Mercury orbiter), STEREO (a pair of solar observers), MRO (Mars Reconnaissance Orbiter), and New Horizons (a Pluto flyby mission). The use of turbo codes saved an additional 1.5 dB over the concatenated RS and convolutional code, enabling operation at about 1 dB from the Shannon limit.

This paper describes recent activities in the next generation of FEC code development and standardization, which has concentrated on low-density parity-check (LDPC) codes. These codes also achieve excellent performance, within about 1 dB of the Shannon limit. While turbo codes are best suited for low code rates, LDPC codes are better at rates higher than 1/2, and have even lower implementation complexities, making them attractive for the ultra-high data rates, up to 1 Gbps, envisioned for near-earth and lunar missions in the next decade.

The effort to standardize the LDPC codes is now underway. In cooperation with the international community, NASA recently completed the Coding, Modulation, and Link Protocol (CMLP) study[4] that analyzed each link in NASA's Space Communication and Navigation (SCaN) architecture, which is a comprehensive plan of NASA spacecraft missions across the solar system through 2030.

The CMLP study evaluated a number of figures of merit to arrive at recommendations for FEC codes. Principal among these is the power efficiency (measured as the ratio between information-bit energy to noise power spectral density, E_b/N_0), and for deep space missions, it dominates all other concerns. For spacecraft at Mars and closer, achievable data rates can be high enough that bandwidth efficiency becomes a concern to satisfy radio spectrum requirements. Bandwidth efficiency is largely achieved through modulation design, but it affects code selection through the code rate. At high data rates, implementation complexity also becomes an issue. Encoders are far simpler than decoders, so it is the decoder complexity that is of interest, and a somewhat imprecise measure is the average number of computations used per decoded bit. Typically there is also a desire to keep codewords short, to satisfy a latency constraint at low data rates, or to simplify interactions with higher-level protocols, or to reduce encoder and decoder complexity and memory requirements. The maturity of hardware, alignment with existing infrastructure support, and the desire to keep the total number of codes small also play roles in the standardization of new codes.

With each of these goals in mind, we briefly review the selection and standardization of turbo codes in section II, and discuss LDPC codes in section III. Implementation considerations for LDPC codes are considered in section IV. Finally, we return to the design goals in section V and see how the various families of codes fit together as a coherent whole.

II. Turbo codes

Coding theorists have traditionally designed error correcting codes with a lot of structure, allowing the use of provably optimum decoders, although the theory says that codes chosen at random should perform well if their block size is large enough. The challenge to find practical decoders for "almost" random, large codes had not been seriously considered until 1993. In this year, the introduction of turbo codes by Berrou

and Glavieux[5] revolutionized the field, and started modern coding theory. Modern codes have relatively little structure so optimum decoding is intractably complex and theoretical analysis is difficult, but experiments show that sub-optimum decoders can perform spectacularly well. On the Additive White Gaussian Noise (AWGN) channel, performance curves within 0.7 dB of the ultimate Shannon limit have become common, and with sufficiently long block lengths, even this gap can be eliminated.[6]

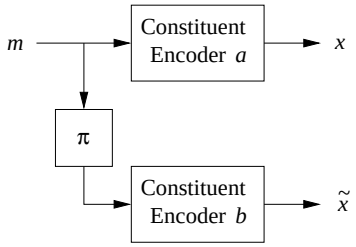

Figure 1. A turbo encoder.

While variations abound, standard turbo codes consist of a parallel concatenation of two recursive convolutional encoders, one of which is fed a permuted (or interleaved) version of the information bits, as shown in figure 1. The convolutional encoders have an Infinite Impulse Response (IIR) and a short constraint length, 16 states apiece in the turbo codes standardized by CCSDS.[3] The interleaver is designed to match low-weight outputs from one constituent encoder with high-weight outputs from the other. This matching challenge has proven the most difficult part of turbo code design, and imperfections in the interleaver result in low weight codewords. These are a primary source of the "error floor", or reduction in slope of the code's Word Error Rate (WER) performance curve at sufficiently high Signal to Noise Ratio (SNR). They are also the source of undetected decoding errors (when the transmitted message decodes to a different codeword), and a turbo code should be concatenated with an outer Cyclic Redundancy Check (CRC) code to detect these.

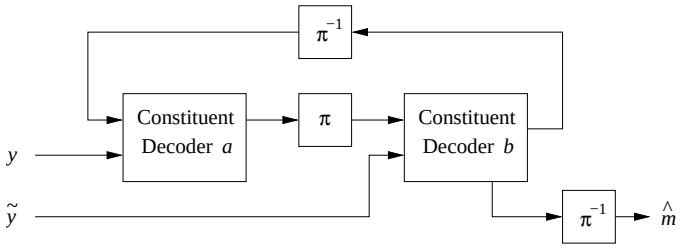

Figure 2. A turbo decoder primarily contains a feedback loop with two modified BCJR decoders, a permutation π, and its inverse.

Turbo decoding uses a suboptimal iterative algorithm shown in figure 2, and these codes earn their name by analogy to the feedback used in automotive turbochargers. The decoder applies a modified Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm[7] to the noisy symbols y from one of the constituent convolutional codes, applies permutation π to the resulting "extrinsic information," combines the result with the noisy symbols $\tilde{y}$ from the other constituent code, inversely permutes the resulting extrinsic information, and repeats. After some number of iterations, an estimate $\hat{m}$ is made of the transmitted message. The number of iterations may be fixed, or a "stopping rule" may be used to reduce the average number of iterations required by halting the decoder when its tentative decisions appear sufficiently certain. The BCJR algorithm is moderately complex and inherently serial, so turbo decoders are more readily implemented in software on a general-purpose processor than in application-specific hardware.

Turbo codes have been included on several spacecraft. A test of turbo codes was performed in late 2004 by SMART-1, a lunar orbiter, but they were not used for significant data return. The MESSENGER spacecraft (a Mercury orbiter launched on August 3, 2004), Mars Reconnaissance Orbiter, and New Horizons (Pluto flyby) are all currently using turbo codes for their primary data return.

III. LDPC code design

LDPC codes were first introduced by Gallager[8] in his 1960 Ph.D. dissertation. Initially, they were prohibitively complex and were subsequently forgotten until a series of papers,[9,10] appeared in the late 1990s and generated renewed interest. While turbo codes are most easily defined via their encoding algorithm, LDPC codes are always defined by their low-density parity check matrices, which have only a few nonzero entries in each row and column. Most LDPC codes are binary, though there is a modest body of research over larger finite fields of characteristic 2. Gallager's original LDPC codes were binary and regular, meaning that each column contained the same number of ones, as did each row.

A sparse parity check matrix can also be described as a bipartite Tanner graph[11] consisting of a *variable node* to represent each matrix column (or degree of freedom), a *check node* to represent each row (or constraint equation), and an edge connecting a variable node to a check node for each nonzero entry in the matrix.

Decoders for LDPC codes use a message-passing algorithm based on the Tanner graph. A simple computation is performed at each variable node to compute soft code symbol estimates, these estimates are passed as messages to the check nodes where more simple computations are performed to compute correction terms, and these correction terms are passed back to the variable nodes. The variable nodes update their estimates, and this process is repeated until a stopping condition is satisfied. The details of the algorithm are well known,[12] and are not repeated here.

Because LDPC codes so strongly emphasize the parity check matrix H instead of the generator matrix G, encoding algorithms are not obvious. As for any block code, one can construct a systematic generator matrix from the appropriate matrix inverse of H. In general, G is dense, so the complexity of encoding by matrix multiplication scales quadratically with the block length.

Research in LDPC code design has aimed to achieve several goals. LDPC codes with irregular node distributions can operate at a lower threshold SNR than regular LDPC codes, particularly at lower code rates, though this comes at the cost of increased decoder computation. The suboptimal iterative decoding algorithm can get lost as it searches for a codeword, and some H matrices have more traps than others do. While a different phenomenon from low-weight codewords, decoder convergence failures also cause error floors in LDPC performance curves rather like those for turbo codes. A very practical consideration is that the connections in a large and randomly organized Tanner graph can become difficult to manage when implementing a decoder at high speed in hardware. Some LDPC codes also have simpler encoders than the dense matrix multiplication described above.

By a variety of paths, many researchers have found that a *protograph*[13] or *base graph*[14] design technique addresses many of these concerns. One starts with a small Tanner graph with just a few nodes, replicates it many times, and interconnects the copies. When the protograph is replicated T times, each edge is replicated into a bundle of T edges, now connecting T variable nodes to T check nodes. The copies of the protograph are interconnected by "unplugging" these edges from their check node sockets, permuting them, and reconnecting them.[15] This process is repeated for each bundle of T edges, or *edge type*.[16]

While the resulting *derived* or *lifted* graph is T times as large as its protograph, it inherits many of the protograph's properties. It has the same code rate (except possibly for coincidental redundancies due to the particular permutations chosen), the same distribution of variable and check node degrees, and even neighborhoods are preserved: nodes in the full graph remain connected to the same mixture of edge types as the nodes in the protograph. These properties mean that full LDPC codes can be designed by applying techniques such as density evolution[17,18] to the protograph. Density evolution allows accurate prediction of a code's threshold SNR, and modest control over its error floor.

Figure 3 shows a regular $(4, 32)$ protograph describing a rate 7/8 code, and figure 4 shows a family of irregular protographs describing codes of rate $(n + 1)/(n + 2) = \{1/2, 2/3, 3/4, 4/5, \ldots\}$. In both figures, filled circles represent variable nodes, circles with crosses in them represent check nodes, and open circles represent variable nodes corresponding to symbols that are *punctured*, i.e., are not transmitted over the channel. Codes derived from the irregular protograph are known as AR4JA codes, because their design was motivated by an Accumulate, Repeat-by-4, Accumulate encoding algorithm, where the second accumulator shown on the right side of the protograph was modified to have a "jagged" appearance.

When a protograph is expanded into a full LDPC code, one must choose permutations for each edge type. Experimentation has shown that while a particularly bad choice can destroy the code's performance, good choices abound. Cyclic shifts have proven adequate, and result in highly structured Tanner graphs well suited to hardware decoders. In matrix form, these appear as cyclicly shifted identity matrices, known as *circulants*. In some cases, this block-circulant structure is preserved through the matrix inversion necessary

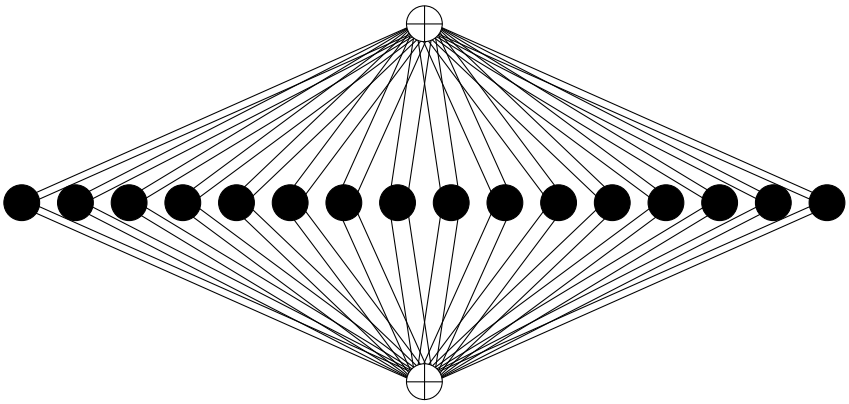

Figure 3. Protograph for a regular (4,32) rate 7/8 code

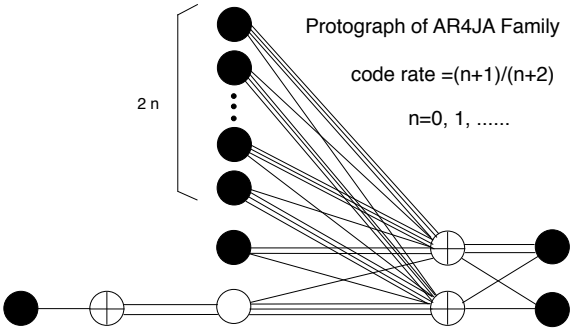

Figure 4. Protographs in the AR4JA family with rates $1/2$ and higher.

to compute a generator matrix. The resulting (dense) block-circulant generator matrices permit simple high-speed encoders in hardware.

Ten codes have been proposed for CCSDS standardization.[19] Nine of these are AR4JA codes, of rates 1/2, 2/3, and 4/5, and blocklengths of 1024, 4096, and 16384 information bits. The tenth code, named C_2 for historical reasons, is of rate approximately 7/8, and is a shortened and extended version of a code built from the protograph in figure 3. Performance curves for each of these codes is shown in figure 5.

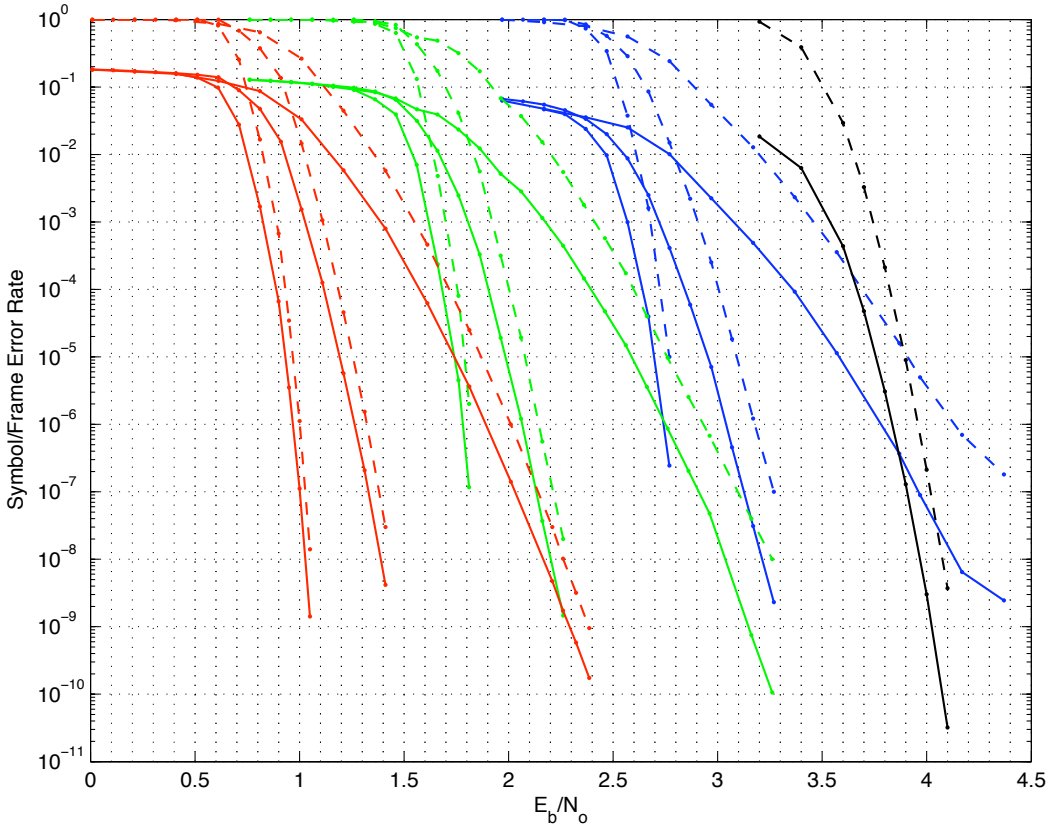

Figure 5. Bit error rate (solid) and codeword error rate (dashed) for nine AR4JA codes and C_2, with code rates 1/2 (red), 2/3, (green), 4/5 (blue), and 0.8752 (black); and blocklengths k=16384, 4096, 1024 (left to right in each group), and 7156 (code C_2).

IV. LDPC decoder implementation

Both code C_2 and the AR4JA family of LDPC codes are built from protographs and circulants, and so are particularly amenable to decoders implemented in hardware, particularly on Field Programmable Gate Arrays (FPGAs). Unlike many LDPC codes, the interconnections between the computation units are well organized, as sketched in figure 6.[20] Because the full Tanner graph is built from many copies of a small protograph, the figure shows variable nodes as groups of circles, each group derived from a single node in the protograph. Similarly, the check nodes are shown as stacked sets of squares. In an FPGA, one computation unit is instantiated for each stack of nodes, and decoding is performed partially in parallel, and partially serially. First, the received soft symbols from the channel are stored in several memories shown across the top of the figure, after being scaled appropriately into log likelihood ratios. Then simultaneously, each variable node unit performs the computation associated with the top node in its stack: it fetches a channel symbol and incoming messages from each of the edge memories (initialized to zero), computes outgoing messages, and writes these back to the edge memories. The variable node computation units repeat this operation sequentially for each node in their stacks. When complete, the check node computations are scheduled similarly. The check node computation nodes simultaneously fetch messages from the edge memories, compute correction terms associated with the first node in each stack,

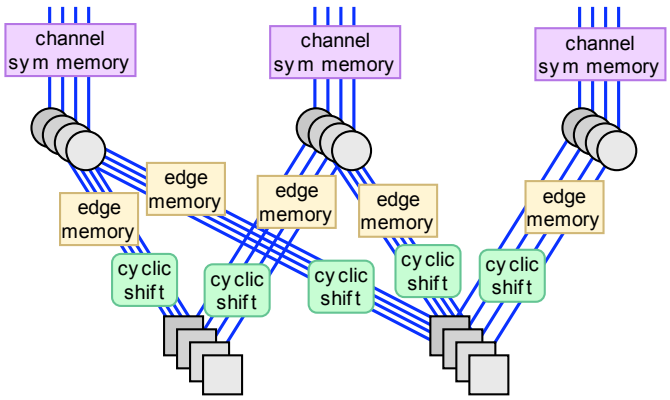

Figure 6. One implementation of an LDPC decoder, capitalizing on the circulant + protograph construction. All the nodes in one copy of the protograph are updated simultaneously, and additional copies are updated serially.

and write these back to the memories. They sequentially perform computations for each subsequent copy of the protograph node, and the entire process is repeated until a stopping condition is satisfied. The cyclic shifts used to permute the edge messages are implemented simply by adding appropriate constant offsets to the edge memory addresses during the check node phase of each iteration. Circular addressing is achieved automatically by the overflow and wrap-around of the binary counters, because the size of each circulant is a power of two.

For a particular code, various decoders can be built by "unfolding" the protograph, thus doubling the decoder speed and hardware requirements as many times as desired. For the AR4JA ($n = 2048, k = 1024$) code, the protograph of figure 4 was expanded to 80 variable nodes (16 of them punctured) and 48 check nodes. This many computation units fill just over half the logic of a Xilinx Virtex2-8000 FPGA, yielding a decoder that runs at 160 Msps (80 Mbps). This decoder, or derivatives of it, may be used by future missions such as the Constellation Program.[21] Alternatively, a smaller and slower decoder may update only a portion of the protograph in each clock cycle. One version runs at 4 Msps and is intended for implementation on the Mars Reconnaissance Orbiter for communication with the Mars Science Lander.

In addition to the LDPC decoder itself, a complete communications system using an LDPC code requires several supporting functions, including a pseudo-randomizer, frame synchronizer, and flow control buffers. Radio systems require sufficiently frequent transitions to maintain symbol synchronization. These LDPC codes are systematic, so messages with long runs of zeros, for example, will encode to sequences with long runs of zeros. This problem is resolved by complementing a pseudo-random subset of the code symbols.[3,22]

LDPC codes are block codes, so when codewords are transmitted as a serial bit stream, the codeword boundaries must be identified before decoding. To permit this, synchronization markers are inserted between codewords, and the receiver can search for this known pattern. While frame synchronization is a classic problem, it is more difficult with LDPC codes because they can operate at a lower symbol SNR than conventional codes. In 1972, Massey derived the optimum frame synchronization algorithm,[23] and also suggested a low-complexity approximation to it. While his algorithms have received little attention, they are now excellent choices for LDPC coded systems.

With an LDPC decoder, one might choose to perform some fixed number of iterations on every codeword. To achieve an error rate close to the decoder's capability, one must set this number of iterations to many times the average required. For example, the rate 1/2 AR4JA code of size ($n = 8192, k = 4096$) achieves WER $\approx 10^{-6}$ with 200 iterations at $E_b/N_0 = 1.35$ dB. Instead, a decoder that stops when a codeword is found (or after a maximum of 200 iterations), requires an average of only 22.5 iterations. These two bounds are shown by the dashed lines in figure 8. Suppose a communications system contains an LDPC decoder that can perform some given number of iterations per second, and noisy codewords arrive every I_{arr} iterations. Then a fixed-iterations decoder can perform only I_{arr} iterations per codeword, and its performance is shown by the $B = 0$ curve in figure 8.

A better system can be built by using input and output buffers and an LDPC decoder that stops early if a codeword is found, determined by verifying all the parity check equations. This system requires less computation per codeword, so it can run faster, or deliver a smaller error rate, or both.

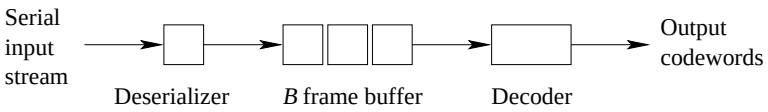

Figure 7. An LDPC decoder coupled with a buffer large enough to hold B noisy codewords.

To demonstrate this, we consider an LDPC decoder coupled with an input buffer that can store B noisy codewords as shown in figure 7. If necessary, an output buffer of the same size can be added to provide a constant-rate decoded bitstream. As each noisy codeword is received, it is placed in the input buffer; if this buffer is full, space is made available by ejecting the incompletely decoded word from the decoder and preempting it with the next noisy codeword in line. The decoder has no limit on the number of iterations it may perform; it iterates until it finds a codeword, or until it is preempted. The performance of this buffered decoder is shown in figure 8 with buffers of size $B = 1$ through $B = 5$.

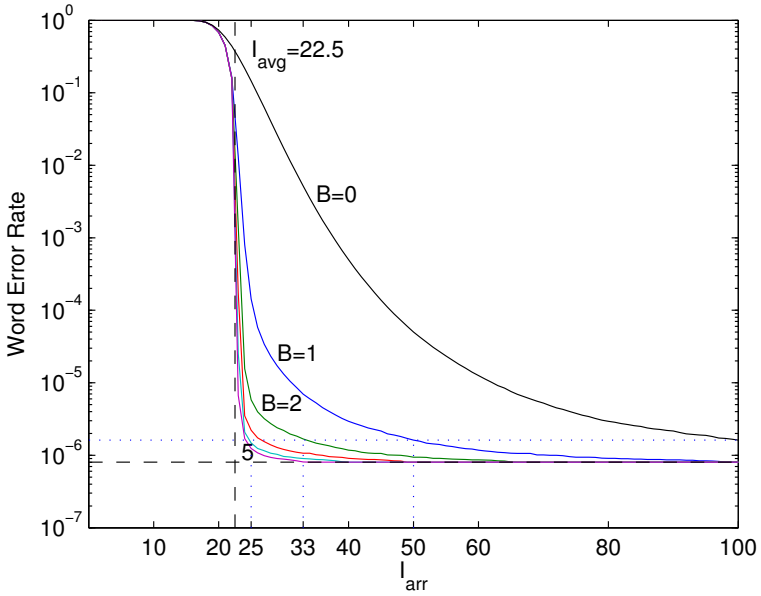

Figure 8. Performance of a variable-iterations decoder with preemptive buffer control as a function of frame inter-arrival time I_{arr}, with buffer sizes (right to left) $B = 0, 1, 2, 3, 4, 5$, for AR4JA $(8192, 4096)$ code, $E_b/N_0 = 1.35$ dB.

If codewords arrive every $I_{\text{arr}} = 100$ iterations, a fixed-iterations decoder achieves WER=1.7×10^{-6}. A variable-iterations decoder with a buffer that holds only $B = 1$ noisy codeword can achieve the same error rate when $I_{\text{arr}} = 50$; with $B = 2$, one can reduce I_{arr} to 33; and with $B = 3$, $I_{\text{arr}} = 25$. That is, at low error rates, a variable-iterations decoder with size B buffer provides the same error rate as a fixed-iterations decoder that runs $B + 1$ times faster.

All of the components described have been implemented in Xilinx FPGAs to characterize algorithm performance, and to demonstrate complete systems. One system test was performed at Johnson Space Center with the Integrated Receiver used with the Space Network (SN) of geosynchronous TDRSS (Tracking and Data Relay Satellite System) orbiters. Another was performed at JPL with a commercial high data rate receiver at 30 Mbps.[22]

V. Code selection and standardization

Recall that in the introduction, we introduced four figures of merit: power efficiency, bandwidth efficiency, information block length, and decoder complexity. Figure 9 shows the achievable trade space between the first two of these, with contours for the third, as determined using Gallager's random coding bound for ensemble averages of codes.[24] Power efficiency, E_b/N_0, is usually expressed in logarithmic units of decibels,

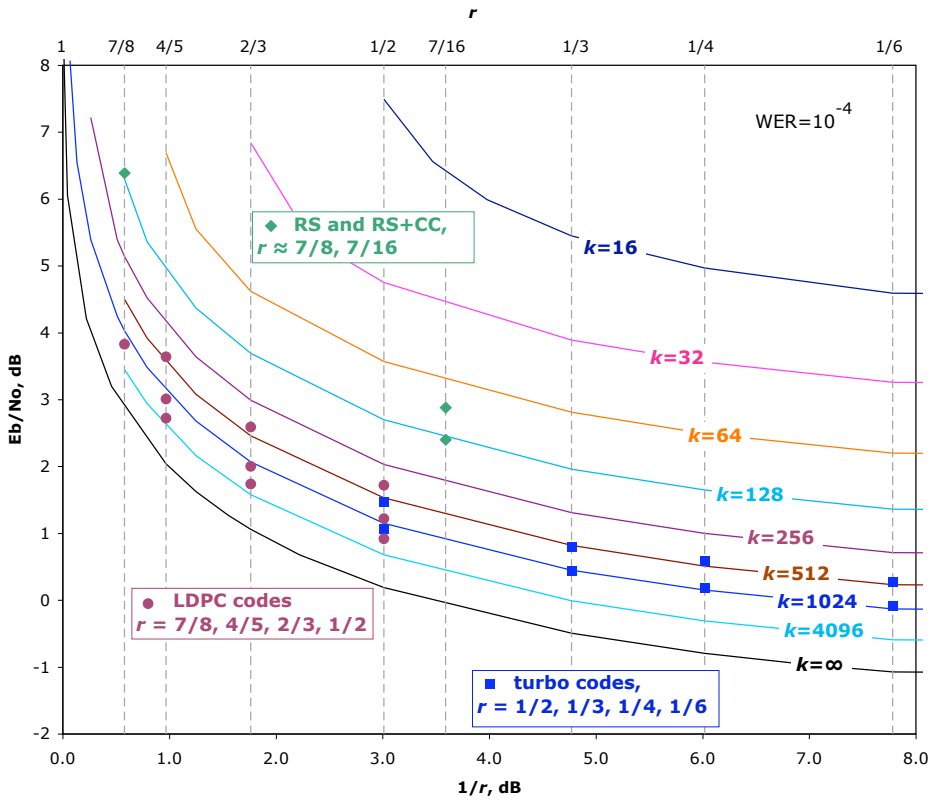

Figure 9. Gallager's random coding bound for the BI-AWGN channel, for various information block sizes, with the CCSDS-standard codes and proposed LDPC codes shown.

and we choose to do the same with the other metrics. Bandwidth efficiency is measured in (bits/second)/Hz, and is inversely proportional to the code rate r. Hence, we have labeled the horizontal axis in units of $10\log_{10}(1/r)$, with a second axis along the top of the figure showing r for convenience. Information block length, which determines latency, is also appropriately measured in logarithmic units, so we have shown curves for block length k in powers of two, though they are labeled in conventional units.

Good codes should lie close to this three-dimensional surface, and a well-chosen set of codes for standardization should populate the surface roughly uniformly over the region of interest. On this figure, points achieved at WER=10^{-4} by the CCSDS-standard turbo codes[3] are shown as squares, with code rates $r = \{1/6, 1/4, 1/3, 1/2\}$ for the longest ($k = 8920$) and shortest ($k = 1784$) blocklengths. Points for the ten proposed LDPC codes[19] are shown as circles. The AR4JA family has rates $r = \{1/2, 2/3, 4/5\}$ and blocklengths $\{k = 1024, 4096, 16384\}$. The tenth code has $r \approx 7/8$ and $k = 7136$. Points for additional CCSDS-standard codes are shown with diamonds. These include the $(n = 255 \times 8, k = 223 \times 8)$ rate $r \approx 7/8$ Reed-Solomon code alone, and also when concatenated with the classic $(7, 1/2)$ convolutional code. The concatenated rate is $r = 0.437$, and the block length is determined by the interleave depth, typically set between $I = 1$ and $I = 5$ to give k between 1784 and 8920 bits.

When a block length of $k = 1784$ information bits or above is acceptable, applications that are particularly constrained by power efficiency typically choose codes of rate 1/2 or below, and turbo codes are suitable. As bandwidth efficiency becomes a greater concern, one moves to code rates of rate 1/2 or above, and LDPC codes serve better. Both options considerably outperform the Reed-Solomon based codes of comparable block length.

When short blocklengths are required, perhaps to meet a latency requirement at a low data rate, one is forced towards the upper right corner of figure 9. Some traditional short block codes are very good when k must be especially small. In the intermediate region, $32 \leq k \leq 512$, LDPC codes may provide new options. Such codes may be useful for severely constrained situations where limited communication is necessary, such as command uplink to spacecraft. Code design for these applications is particularly challenging because an extremely low undetected error rate is desired. Study of new codes in this region by CCSDS member

agencies is now beginning.

The fourth figure of merit, decoder complexity, is more difficult to quantify. Among LDPC codes, one can count the average number of message computations required per decoded information bit, IE/k, where I is the average number of iterations required and E is the number of edges in the Tanner graph. Experimentally, one finds that this is a function of the symbol SNR E_s/N_0, the block length k, the Word Error Rate (WER), and the design of the LDPC code.[25] When the last of these is isolated from the other factors, the AR4JA codes are about twice as complex to decode as Gallager's original regular $(3, 6)$ LDPC codes. In exchange for this complexity, the AR4JA design reduces the required SNR by about 0.5 dB relative to the $(3, 6)$ codes, when code rate, block length, and WER are compensated for. This trade-off is shown in figure 10 where complexity is expressed logarithmically in dB, with triangles marking points for several AR4JA codes, asterisks for a couple Gallager-$(3, 6)$ codes, and additional symbols for members of other LDPC code families. While the AR3A code family, a precursor to the AR4JA family, appears to outperform the others in this figure, it also suffers from unusually high error floors. Especially for spacecraft applications with the decoder located on the ground, SNR is a more significant metric than decoder complexity, so the AR4JA family is an attractive choice.

The particular blocklengths used for the family of turbo codes were chosen to match those of the Reed-Solomon codes, and experience is showing that this is of limited benefit. Instead, for the LDPC codes, exact powers of two have been chosen. This is convenient for microprocessor systems, and means that the code rates are exact ratios of small integers which can simplify the necessary clock generation in hardware. For the various code rates, the number of information bits, k, is held constant rather than the number of code symbols, n. This allows partitioning an information stream into codewords without knowing the code rate that will be used to transmit the data.

Both the turbo and LDPC codes are systematic, to simplify system testing in a noiseless environment. The turbo codes typically exhibit error floors at a Word Error Rate between 10^{-5} and 10^{-6}; if the LDPC codes have error floors, they are so far down that their locations are unknown.

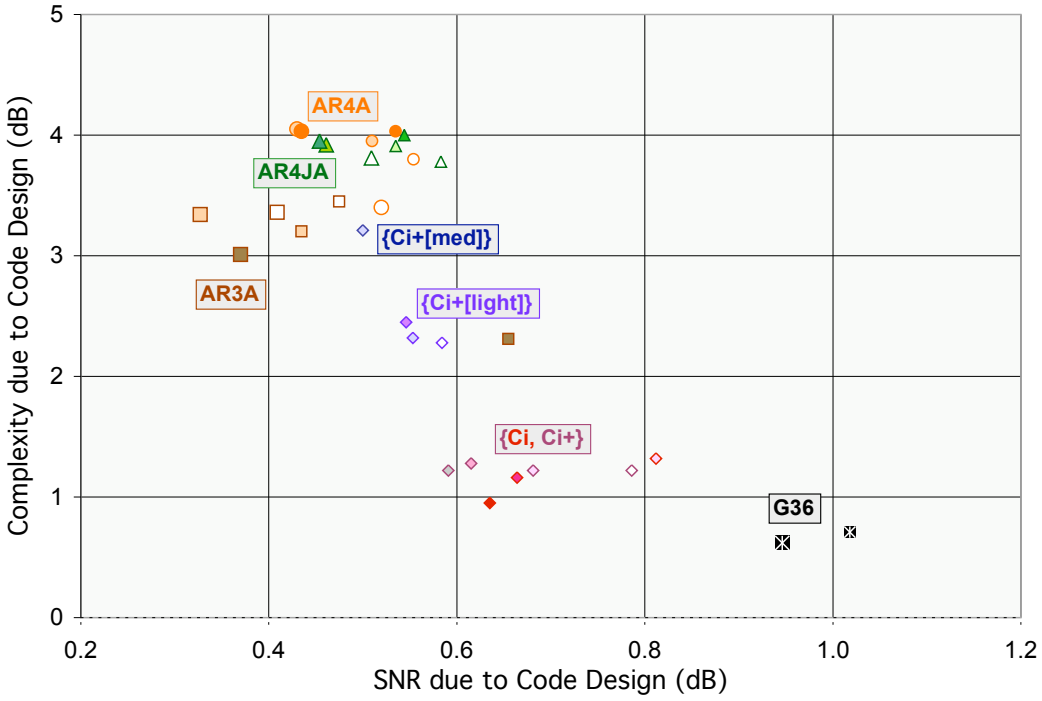

Figure 10. Tradeoff between normalized decoder complexity and size-constrained non-optimality among several LDPC code families at a codeword error rate of 10^{-4}.

VI. Conclusions

The Consultative Committee for Space Data Systems (CCSDS) selects and standardizes error correcting codes for space applications. It is difficult to predict specific future needs, so it is more practical to provide a "toolbox" containing enough codes to cover the design space of interest for spacecraft missions. Simultaneously, the number of codes should be kept reasonably small to limit implementation and support costs.

When selecting an error correcting code, spacecraft mission designers have a variety of competing goals. Power efficiency is perhaps the most obvious, and for deep space missions, it dominates all other concerns. For high data-rate applications, bandwidth efficiency is important, and this can be achieved by using higher order modulations and spectrally efficient pulse shapes, as well as by increasing the code rate. Some applications, particularly voice on low data-rate links, have a latency constraint that limits the block lengths that can be used. Decoder complexity is also a concern, especially when the decoder is onboard a spacecraft, but also for earth-based decoders at high data rates.

Modern turbo and LDPC codes considerably outperform traditional Reed-Solomon, convolutional, and other codes, and specific codes have been optimized for the constraints of spacecraft use. Turbo codes had a head-start, so they have been incorporated into CCSDS standards and are in use on several current missions. LDPC codes are now described in draft standards, experimental hardware has been built and tested, and their use is planned for a variety of upcoming spacecraft.

References

[1]Andrews, K. S., Divsalar, D., Dolinar, S., Hamkins, J., Jones, C. R., and Pollara, F., "The Development of Turbo and LDPC Codes for Deep-Space Applications," *Proceedings of the IEEE*, Vol. 95, No. 11, Nov. 2007, pp. 2142–2156.

[2]Collins, O., Dolinar, S., McEliece, R., and Pollara, F., "A VLSI Decomposition of the deBruijn Graph," *Journal of the ACM*, Vol. 39, No. 4, 1992, pp. 931–948.

[3]The Consultative Committee for Space Data Systems (CCSDS), *Telemetry Channel Coding (101.0-B-4 Blue Book)*, CCSDS, Newport Beach, CA, May 1999.

[4]Deutsch, L. and et al., F. S., "Coding, Modulation, and Link Protocol (CMLP) Study Final Report," NASA Space Communications and Navigation (SCaN), Jan. 2008.

[5]Berrou, C., Glavieux, A., and Thitimajshima, P., "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes," *IEEE International Conference on Communications*, May 1993, pp. 1064–1070, Geneva, Switzerland.

[6]Chung, S.-Y., Forney, Jr., G. D., Richardson, T. J., and Urbanke, R., "On the Design of Low-Density Parity-Check Codes within 0.0045 dB of the Shannon Limit," *IEEE Communications Letters*, Vol. 5, No. 2, Feb. 2001, pp. 58–60.

[7]Bahl, L., Cocke, J., Jelinek, F., and Raviv, J., "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transactions on Information Theory*, March 1974, pp. 284–287.

[8]Gallager, R., *Low-Density Parity-Check Codes*, Ph.D. thesis, Mass. Inst. Tech., Cambridge, Sept. 1960.

[9]Mackay, D., "Good error correcting codes bases on very sparse matrices," *IEEE Transactions on Information Theory*, Vol. 45, March 1999, pp. 399–431.

[10]MacKay, D., Wilson, S., and Davey, M., "Comparison of constructions of irregular Gallager codes," *IEEE Transactions on Communications*, Vol. 47, No. 10, Oct. 1999, pp. 498–519.

[11]Tanner, R. M., "A Recursive Approach to Low Complexity Codes," *IEEE Transactions on Information Theory*, Sept. 1981, pp. 533–547.

[12]Kschischang, F. R., Frey, B. J., and Loeliger, H.-A., "Factor Graphs and the Sum-Product Algorithm," *IEEE Transactions on Information Theory*, Feb. 2001, pp. 498–519.

[13]Thorpe, J., "Low-Density Parity-Check (LDPC) Codes Constructed from Protographs," IPN Progress Report 42-154, JPL, Aug. 2003.

[14]Lin, S., "Quasi-Cyclic LDPC Codes," CCSDS working group white paper, Oct. 2003.

[15]Thorpe, J., Andrews, K., and Dolinar, S., "Methodologies for Designing LDPC Codes Using Protographs and Circulants," *IEEE International Symposium on Information Theory*, June 2004, p. 238.

[16]Richardson, T., "Multi-Edge Type LDPC Codes," Presented at the Workshop honoring Prof. Bob McEliece on his 60th birthday (not in proceedings), California Insitute of Technology, Pasadena, CA.

[17]ten Brink, S., "Convergence of Iterative Decoding," *Electronics Letters*, Vol. 35, May 1999, pp. 806–808.

[18]El Gamal, H. and Hammons, Jr., A. R., "Analyzing the turbo decoder using the Gaussian approximation," *IEEE Transactions on Information Theory*, Vol. IT-47, Feb. 2001, pp. 671–686.

[19]The Consultative Committee for Space Data Systems (CCSDS), "Low Density Parity Check Codes for Use in Near-Earth and Deep Space Applications (131.1-O-1 Orange Book)," Aug. 2006.

[20]Jones, C., Dolinar, S., Andrews, K., Divsalar, D., Zhang, Y., and Ryan, W., "Functions and Architectures for LDPC Decoding," *IEEE Information Theory Workshop*, Lake Tahoe, California, Sept. 2007, pp. 577–583.

[21]NASA HQ, "NASA Names New Crew Exploration Vehicle Orion," Press Release, Aug. 2006.

[22]Cheng, M. K., Lyubarev, M., Nakashima, M. A., Andrews, K. S., and Lee, D., "Integrated Performance of Next Generation High Data Rate Receiver and AR4JA LDPC Codec for Space Communications," *SpaceOps 2008*, May 2008, Heidelberg, Germany.

[23]Massey, J. L., "Optimum Frame Synchronization," *IEEE Transactions on Communications*, April 1972, pp. 115–119.

[24]Dolinar, S., Divsalar, D., and Pollara, F., "Code Performance as a Function of Block Size," TMO Progress Report 42-133, JPL, May 1998.

[25]Dolinar, S. and Andrews, K., "Performance and Decoder Complexity Estimates for Low-Density Parity-Check (LDPC) Code Families," IPN Progress Report 42-168, JPL, Feb. 2007.